Available online at www.sciencedirect.com

ScienceDirect

Journal of Discrete Algorithms 5 (2007) 262–279

JOURNAL OF DISCRETE ALGORITHMS

www.elsevier.com/locate/jda

Cut problems in graphs with a budget constraint ☆

Roee Engelberg [a,*], Jochen Könemann [b,1], Stefano Leonardi [c,2], Joseph (Seffi) Naor [a,2]

[a] *Computer Science Department, Technion, Haifa 32000, Israel*

[b] *Department of Combinatorics and Optimization, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada*

[c] *Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy*

Received 8 August 2005; accepted 11 May 2006

Available online 21 July 2006

Abstract

We study budgeted variants of classical cut problems: the *Multiway Cut* problem, the *Multicut* problem, and the *k-Cut* problem, and provide approximation algorithms for these problems. Specifically, for the budgeted multiway cut and the k-cut problems we provide constant factor approximation algorithms. We show that the budgeted multicut problem is at least as hard to approximate as the sparsest cut problem, and we provide a bi-criteria approximation algorithm for it.

Keywords: Approximation algorithms; Budget problems; Graph theory; Cut problems; Combinatorial optimization

1. Introduction

Given an undirected graph $G = (V, E)$ with a positive cost function on the edges $c : E \to \mathbb{Q}^+$, and a subset of vertices $S \subseteq V$, called *terminals*, the well-known *multiway cut* problem is to find a minimum cost subset of edges whose removal disconnects the terminals from each other. The study of the multiway cut problem was initiated by Dahlhaus, Johnson, Papadimitriou, Seymour and Yannakakis [1], who proved that it is MAX-SNP-hard even when restricted to instances with 3 terminals and unit edge cost. They also gave a $(2 - \frac{2}{k})$-approximation algorithm for the problem, where $|S| = k$. Their algorithm finds, for each terminal s_i, a minimum cost cut separating s_i from the remaining terminals, and outputs the union of the $k - 1$ cheapest of the k cuts.

In [2], Călinescu, Karloff and Rabani introduced a $(1.5 - \frac{1}{k})$-approximation algorithm, where $|S| = k$. They considered a linear programming relaxation for the multiway cut problem which embeds the given graph into the $(k - 1)$-dimensional simplex. The algorithm of [2] rounds an optimal solution to the linear programming relaxation; its bound was later improved to ∼1.3438 by [3].

☆ A preliminary version of this paper appeared in Proceedings of the 7th Latin American Theoretical Informatics Symposium (LATIN'06).

* Corresponding author. Tel.: +972-4-829-4934; fax: +972-4-829-3900.

E-mail addresses: roee@cs.technion.ac.il (R. Engelberg), jochen@math.uwaterloo.ca (J. Könemann), leon@dis.uniroma1.it (S. Leonardi), naor@cs.technion.ac.il (J.S. Naor).

[1] This work was done while being on leave at the Dipartimento di Informatica e Sistemistica at Università di Roma "La Sapienza", Italy.

[2] Research partially supported by MIUR-FIRB project Israel and Italy n. RBIN047MH9.

1570-8667/$ – see front matter
doi:10.1016/j.jda.2006.05.002

In this paper we study two budgeted variants of the multiway cut problem that differ in their objective function. In the budgeted variants, given an instance of the multiway cut problem together with an additional positive integer B, the *budget*, the problem is to find a subset of edges whose cost is within the given budget and whose removal maximizes the value of the given objective function.

We say that a pair of terminals (s_i, s_j) is *separated* if there is no path between s_i and s_j, and that a terminal s_i is *isolated* if there is no path between s_i and any other terminal. The number of *isolated* terminals is the objective function of the first *budgeted* variant of the multiway cut problem, referred to as the *budgeted isolating multiway cut* (*BIMC*) problem. In the second *budgeted* variant, referred to as the *budgeted separating multiway cut* (*BSMC*) problem, the objective function is the number of *separated* pairs of terminals. We also consider the *weighted* versions of both *BSMC* and *BIMC*.

An application of the weighted BSMC problem is network design against denial-of-service attacks in networks. In [4], Aura, Bishop and Sniegowski suggest a formal framework for the study of the single-server inhibition attack, which is a common scenario for modelling a denial of service attack. One of the problems they consider is finding the best attack whose cost is within a given budget constraint. In this problem, every client has a non-zero weight denoting its importance. The cost of an attack is the total cost of the disconnected links in the network, and the *value* of the attack is the total weight of the clients separated from the given server. This problem can be considered as a weighted BSMC by setting the weight of every (server, client) pair to be the client's weight.

A well known generalization of the multiway cut problem is the *multicut* problem, which is the problem of finding a minimum cost cut separating a given set of source-sink pairs of vertices. Indeed, the multiway cut problem is a special case of the multicut problem in which the set of source-sink pairs consists of all the pairs of a given set of terminals. Consider the following budgeted variant of the multicut problem. Given is a set of source-sink pairs of vertices together with a budget. Let the source-sink pairs be associated with a non-negative weight. The goal is to find a cut whose cost is within the budget that separates a maximum weight set of source-sink pairs. Thus, this budgeted multicut problem is precisely the weighted version of the BSMC problem.

Finally, given an undirected graph, we consider the problem of finding a set of edges whose cost is within a given budget and whose removal partitions the graph into a maximum number of connected components. This problem, referred to as the *budgeted graph disconnection* (*BGD*) problem, can be thought of as the budgeted version of the *k-cut* problem. In the k-cut problem, an integer k is given and the goal is to find a minimum cost edge set whose removal partitions the graph into at least k connected components. We note that the cardinality version of BGD (in which all the edges have a unit cost) was introduced by Frederickson and Solis-Oba [5], where it was referred to as *the Maximum Components* problem.

1.1. Our results

The hardness of the multiway cut problem implies that both BIMC and BSMC cannot be efficiently solved unless $P = NP$. Although the problem definitions of BIMC and BSMC are closely related, they capture different aspects of the theory of cuts, and therefore differ in their level of hardness. Thus, we study each of the problems independently.

BIMC and weighted BIMC: We give constant factor approximation algorithms that match some of the lower bounds we prove. Our algorithms basically use a greedy approach. In the weighted case we improve on the greedy approach by using an FPTAS for the *knapsack* problem.

Weighted BSMC/Budgeted Multicut: We show that weighted BSMC is at least as hard to approximate as the *Sparsest Cut* problem is (up to a constant). For the sake of comparison, we note the recent series of results regarding the sparsest cut problem initiated by Arora, Rao and Vazirani [6] improving on previous $O(\log k)$-approximations [7,8]. In [6], a new structural theorem about metric spaces of negative type is proved, and an $O(\sqrt{\log n})$ approximation is presented for the uniform case. Chawla, Gupta and Räcke [9] gave an $O(\log^{3/4} k)$-approximation for the general sparsest cut problem, while the current best known result is an $O(\sqrt{\log k} \log\log k)$-approximation due to Arora, Lee and Naor [10].

We notice that the weighted BSMC on *trees* is a special case of the *maximum coverage problem*, and hence it can be approximated using the algorithm of Khuller, Moss and Naor [11]. We provide an analysis of their algorithm's performance with respect to the optimal *fractional* solution of a natural linear programming

relaxation. We then consider the weighted BSMC problem on general graphs. We show that the same relaxation has an unbounded integrality gap, and achieve a bi-criteria approximation of $(\frac{e}{e-1}, O(\log^2 n \log\log n))$ using a recent hierarchical decomposition of graphs by Räcke (see [12] and [13]).

Interestingly, we show that BSMC is related to the budgeted variant of the *Sparsest Cut* problem. Specifically, we prove that for certain weight functions, an approximation algorithm for BSMC can be used to derive an approximation algorithm for the budgeted sparsest cut problem, and vice versa.

BGD: We give a constant factor approximation algorithm for BGD which is a generalization of Frederickson and Solis-Oba's algorithm [5] for the cardinality version of BGD. The analysis we present for our algorithm is based on the Gomory–Hu tree (see [14]) and relies upon the approximation algorithm of Saran and Vazirani [15] for the k-cut problem.

1.2. Related work

To the best of our knowledge, except for the cardinality version of BGD, all of the above mentioned *budgeted* cut problems are studied for the first time here. Nevertheless, there is a vast literature on budgeted optimization problems and we mention the following relevant works.

The *k-median* problem is a fundamental problem in which one has to minimize the connection cost of *cities* to opened *facilities*, while only k facilities can be opened. The constraint on the number of opened facilities is the *budget* constraint. In the *Lagrangian relaxation* of the k-median problem the budget constraint is relaxed by moving it into the objective function, i.e., the constraint on the number of opened facilities is replaced by a cost for opening a facility. This is a special case of the *facility location* problem. Some approximation algorithms for the k-median problem (for example, see [16]) exploit known approximation algorithms for the facility location problem using Lagrangian relaxation.

In [17], Naor, Shachnai and Tamir introduce a general approximation technique via Lagrangian relaxation for a class of *subset selection* problems, which is a class of budget problems. They apply their technique to problems of *real-time scheduling with budget*. They also show that, for some of these problems, the greedy approach yields a constant factor approximation algorithms.

Vohra and Hall [18] considered a budgeted variant for the classical *set cover* problem, while Khuller, Moss and Naor [11] studied its weighted variant. Khuller et al. gave a constant factor approximation algorithm for the problem that is based on the greedy approach, and showed that their result is tight under a (weak) assumption on the hardness of *NP*. Their result points out the possible gap between the hardness of a problem and the hardness of its budgeted variant, as the *set cover* problem cannot be approximated within a factor of $(1-\epsilon)\ln n$ for any $\epsilon > 0$ under the same assumption on the hardness of *NP*. By improving a former work by Wolsey [19], Sviridenko [20] generalized the result of Khuller et al. for the problem of maximizing any submodular function subject to a budget constraint. We note that this framework does not capture most of the problems we deal with in this paper, but it does capture the weighted BSMC on *trees*.

Recently, two variants of a budgeted cut problem were studied. In the *minimum size bounded capacity cut* (respectively, *maximum size bounded capacity cut*), given is a graph $G = (V, E)$ with edge costs and vertex weights, a source $s \in V$, a sink $t \in V$, and a budget B. The purpose is to find an $s-t$-cut (S, T) whose cost is at most B such that the total weight of the vertices in S is minimized (respectively, maximized). Hayrapetyan, Kempe, Pál and Svitkina [21] presented an efficient $(\frac{1}{1-\lambda}, \frac{1}{\lambda})$-bi-criteria approximation algorithm for any $0 < \lambda < 1$ for the minimization variant, while Svitkina and Tardos [22] studied the maximization variant to which they introduced an $(1, \log^2 n)$-bi-criteria approximation.

1.3. Organization

The rest of this paper is organized as follows. In Section 2 we formally define the problems considered in this paper. The BIMC problem is studied in Section 3, while BSMC is studied in Section 4. Section 5 deals with the BGD problem. We conclude in Section 6 with further discussion.

2. Preliminaries

In this section we formally define the problems considered in this paper. In all of these problems, we are given an undirected graph $G = (V, E)$ with a positive cost function on the edges $c : E \to \mathbb{R}^+$, and a positive *budget* B.

Problem 1 *(Budgeted Graph Disconnection (BGD)).* Find a subset of edges $C \subseteq E$ of cost at most B whose removal partitions the graph into the maximum number of connected components.

Other problem definitions are based on the following terms.

Definition 2 *(Separation).* Given a subset of edges $C \subseteq E$, we say that vertices s and s' ($s' \neq s$) are *separated by* C, or, equivalently, that C *is a separating cut of* (s, s'), if every path between s and s' contains *at least* one edge from C.

Definition 3 *(Isolation).* Let $S \subseteq V$ be a given subset of vertices. Given a subset of edges $C \subseteq E$, we say that a vertex $s \in S$ is *isolated by* C, or equivalently, that C *is an isolating cut of* s, if for every $s' \in S$, $s' \neq s$, s and s' are separated by C.

In the following problems, we are additionally given a subset of vertices $S \subseteq V$ (let $k = |S|$), called *terminals*. In the weighted BIMC problem we are also given a weight function on the terminals, $w : S \to \mathbb{Z}^+$, used in the next definition.

Definition 4. Given a subset of edges $C \subseteq E$, its *isolation weight*, denoted by $w(C)$, is the sum of the weights of the terminals isolated by C.

Problem 5 *(Weighted Budgeted Isolating Multiway Cut (weighted BIMC)).* Find a subset of edges $C \subseteq E$ of cost at most B whose isolation weight is maximized.

Without loss of generality we assume that there exists $s \in S$ such that the cost of the minimum cost isolating cut of s is at most B. We denote by *BIMC* the special case of weighted BIMC where $w(s) = 1$ for every $s \in S$.

In the weighted BSMC problem we are additionally given a weight function on the pairs of terminals, $w : S \times S \to \mathbb{Z}^+$, used in the next definition.

Definition 6. Given a subset of edges $C \subseteq E$, its *separation weight*, denoted by $w(C)$, is the sum of the weights of the pairs of terminals separated by C.

Problem 7 *(Weighted Budgeted Separating Multiway Cut (weighted BSMC)).* Find a subset of edges $C \subseteq E$ of cost at most B whose separation weight is maximized.

Without loss of generality we assume that for every pair $s, s' \in S$, the cost of the minimum cost separating cut of s and s' is at most B. We denote by *BSMC* the special case of weighted BSMC where $w(s, s') = 1$ for every $s, s' \in S$.

With respect to the same input, we define the Sparsest Cut problem.

Definition 8. Given a non-empty subset of vertices $U \subset V$, the cut *associated with* U, denoted by $(U, \overline{U})$, is $\{e = (u, v) \in E : u \in U, v \notin U\}$.

Definition 9. The *Sparsity* of the cut $(U, \overline{U})$ is given by $\frac{c(U,\overline{U})}{w(U,\overline{U})}$, where $w(\cdot)$ is the separation weight.

Problem 10 *(Sparsest Cut).* Find a non-empty subset of vertices $U \subset V$ such that the sparsity of its associated cut is minimized.

Definition 11 *(Bi-criteria approximation for a budget problem).* An algorithm ALG is a *bi-criteria approximation with parameters* (α, β) for a given maximization budget problem Π, or simply an (α, β)*-approximation for* Π, if for

every instance of Π with budget B, *ALG* outputs a solution whose value is at least $|OPT|/\alpha$ and whose cost is at most βB, where $|OPT|$ is the value of the optimal solution with respect to the *given budget* B.

3. The budgeted isolating multiway cut problem

In this section, we study the BIMC and weighted BIMC problems. First we introduce some hardness results, including integrality gaps of two possible linear relaxations. These integrality gaps suggest that an approximation algorithm which is based on these linear relaxations cannot outperform the constant factor approximation algorithm we give for BIMC. Lastly, we also give two approximation algorithms for weighted BIMC, the second of which matches one of the lower bounds we introduce.

3.1. Hardness results

Proposition 12. *Unless $P = NP$, there is no α-approximation for the BIMC problem for all $\alpha > 1/3$.*

Proof. Assume to the contrary that there exists an α-approximation algorithm for the BIMC problem, $\alpha > 1/3$, and denote it by *ALG*. We show how to solve the multiway cut problem with $k = 3$, which is MAX-SNP-hard. Given an instance of the multiway cut problem, let C be the cost of a minimum multiway cut. Notice that for every budget $B \geqslant C$, *ALG* will return a solution that isolates at least $\alpha \cdot 3 > 1$ terminals. Since every cut that isolates at least two terminals isolates all three terminals, it follows that if $B \geqslant C$, *ALG* will isolate all the terminals, and otherwise it will isolate at most one terminal. Thus, by using *ALG*, it is possible to binary search the range $[0, \sum_{e \in E} c(e)]$ for the value C, and furthermore, one can find a minimum multiway cut of the given instance. □

Proposition 12 can be easily generalized as follows (proofs are omitted).

Proposition 13. *Unless $P = NP$ there is no α-approximation for the BIMC problem with k (fixed) terminals, for every $\alpha > 1 - 2/k$.*

Proposition 14. *Unless $P = NP$ there is no α-approximation for the BIMC problem for every $\alpha > 1 - 2/OPT$, where $OPT > 2$ is the number of isolated terminals in an optimal solution.*

3.1.1. Integrality gap of linear programming relaxations

We consider two natural linear programming relaxations for the BIMC problem. In these relaxations we assume that for every $s \in S$, the cost of the minimum cost isolating cut of s is at most B (if not, a slight modification can be made in the relaxations and the relevant claims still hold). The first one is a straight forward formulation of the problem. We assign an indicator variable y_e for every edge $e \in E$, which will be set to 1 iff the edge e is picked to the solution. The budget constraint can be stated accordingly. We also assign a variable x_s for every terminal $s \in S$, which indicates whether terminal s is isolated by the given solution. In order to enforce that a terminal s is isolated if $x_s = 1$, we state the constraint $x_s \leqslant \sum_{e \in P_{s,s'}} y_e$ for each path between s and any other terminal s'.

(N-ISO-LP)

$$
\begin{aligned}
&\max \ \textstyle\sum_{s \in S} x_s && \\
&\text{s.t.} && \\
&\quad x_s - \textstyle\sum_{e \in P_{s,s'}} y_e \leqslant 0 && \text{for every } s, s' \in S \ (s \neq s') \\
&\qquad && \text{and path } P_{s,s'} \text{ from } s \text{ to } s' \\
&\quad \textstyle\sum_{e \in E} c(e) \cdot y_e \leqslant B && \\
&\quad 0 \leqslant x_s \leqslant 1 && \text{for every } s \in S \\
&\quad 0 \leqslant y_e && \text{for every } e \in E
\end{aligned}
$$

Proposition 15. *The integrality gap of N-ISO-LP is at least* 2.

Proof. Consider a star with N leaves that are all terminals, set $B = N/2$, and $c(e) = 1$ for every edge e. An optimal integral solution picks $N/2$ edges and has a value of $N/2$, while an optimal fractional solution is: $y_e = \frac{1}{2}$ for every $e \in E$ and $x_s = 1$ for every $s \in S$. This solution has value N. □

1: **for** each $s \in S$ **do**
2: Find a minimum cost isolating cut for s, and denote it by C_s.
3: **end for**
4: Sort the cuts in a non-decreasing order of their cost.
5: Choose the maximal sequence of cuts, starting from the cheapest, whose total cost is at most B.

Algorithm 1. A greedy algorithm for BIMC.

The second linear programming formulation we consider is derived from the linear programming relaxation of the multiway cut problem presented in [2]. We assume that $S = \{s_1, \ldots, s_k\}$, and embed the given graph into the k-dimensional simplex. We reserve the 0-coordinate for the connected component that contains all the terminals *not* isolated by the solution, and the ith coordinate for the connected component that contains terminal s_i, if terminal s_i is isolated by the solution. Thus, we allow terminal s_i to be mapped to either the 0th component, or the ith component.

$$\begin{aligned}
&\max \textstyle\sum_{s_i \in S} x^i_{s_i} && && \text{(CKR-ISO-LP)}\\
&\text{s.t.}\\
&\quad x^i_{s_i} + x^0_{s_i} = 1 && \text{for } 1 \leqslant i \leqslant k\\
&\quad \textstyle\sum_{i=0}^{k} x^i_v = 1 && \text{for every } v \in V \setminus S\\
&\quad x^i_v \geqslant 0 && \text{for every } v \in V \text{ and } 0 \leqslant i \leqslant k\\
&\quad y_e = \tfrac{1}{2} \textstyle\sum_{i=0}^{k} |x^i_u - x^i_v| && \text{for every } e = (u, v) \in E\\
&\quad \textstyle\sum_{e \in E} c(e) \cdot y_e \leqslant B
\end{aligned}$$

Proposition 16. *The integrality gap of CKR-ISO-LP is at least* 2.

Proof. Consider a clique of N terminals, let $B = N - 1$, and $c(e) = 1$ for every edge. An optimal solution can isolate only one terminal, while an optimal fractional solution is: $x^i_{s_i} = \frac{2}{N}$ for every $1 \leqslant i \leqslant k$ (due to feasibility, the rest of the solution is uniquely defined), which has a value of 2. □

The integrality gaps shown above suggest that using "natural" linear relaxations, one cannot improve on the approximation factor achieved by the following approximation algorithm for BIMC.

3.2. A greedy approximation algorithm for BIMC

The following greedy algorithm for BIMC is a variant of the algorithm presented in [1] for the multiway cut problem. Note that a minimum cost isolating cut for $s_i \in S$ can be computed efficiently by merging the terminals in $S \setminus \{s_i\}$ into a single node r and computing a minimum cut separating r from s_i.

Lemma 17. *Let ℓ denote the number of isolated terminals in an optimal solution. Algorithm 1 achieves an approximation factor of $\frac{1}{2}$ if ℓ is even, and $\frac{1}{2} - \frac{1}{2\ell}$ if ℓ is odd.*[3]

Proof. Let OPT be an optimal solution, and let I denote the set of terminals isolated by OPT. We assume without loss of generality that there is no edge in OPT that can be removed without changing the set of isolated terminals. Let $G' = (V, E \setminus OPT)$. For $s \in I$, let OPT_s be the edges in OPT that have an endpoint in the connected component of s in G'.

Consider the following charging scheme for the terminals in I. Charge the cost of every edge $e \in OPT$ as follows: if there exist two distinct terminals $s \in I$ and $s' \in I$, such that $e \in OPT_s$ and $e \in OPT_{s'}$, then charge each of the two terminals with $c(e)/2$; otherwise, charge the terminal $s \in I$, such that $e \in OPT_s$, with $c(e)$. Denote by $c(s)$ the total cost charged to terminal s. Obviously, since every edge in OPT is clearly paid for by the charging scheme,

$$\sum_{s \in I} c(s) = c(OPT) \leqslant B.$$

[3] For the trivial case in which $\ell = 1$ the algorithm finds an optimal solution.

Since OPT_s is an isolating cut for each $s \in I$,

$$c(C_s) \leqslant c(OPT_s) \leqslant 2c(s).$$

Let A_ℓ be the set of the first ℓ terminals sorted in Step 4 of the algorithm. Notice that

$$\sum_{s \in A_\ell} c(C_s) \leqslant \sum_{s \in I} c(C_s) \leqslant 2 \sum_{s \in I} c(s) \leqslant 2B.$$

Thus, the cost of the first $\lfloor \ell/2 \rfloor$ terminals is at most B, and the lemma follows immediately. □

The above analysis is tight as the following example shows.

Example 18. Let N be an odd integer, let $B = N(N-1)/2$, and consider a graph with the following two connected components:

- A clique of N terminals with $c(e) = 1$ for every edge in the clique;
- A star with N leaves, all of which are terminals, and each leaf is connected to the root by an edge of cost $c(e) = N - 1 - \epsilon$.

Choosing all the clique edges results in a solution whose value is N, while Algorithm 1 chooses only edges from the star, and achieves a value of at most $\lfloor B/(N-1-\epsilon) \rfloor = \lfloor N/2 \rfloor = (N-1)/2$.

3.3. Approximation algorithms for the weighted BIMC problem

We present two algorithms for the weighted BIMC problem.

3.3.1. A greedy algorithm

The following is a generalization of Algorithm 1.

Lemma 19. *Algorithm 2 achieves an approximation factor of* $\frac{1}{4}$.

Proof. We follow the proof of Lemma 17 and only specify the changes needed in the analysis. Let ℓ be the isolation weight of *OPT*, i.e., the value of the optimal solution. By applying the charging scheme, and since the sequence $\{C_i\}_{i=1}^k$ is sorted with respect to the ratio of cost to weight, any prefix of the sequence with isolation weight at most ℓ costs at most $2B$, and similarly, any prefix with isolation weight $\ell/2$ costs at most B. Thus, the cut $\bigcup_{i=1}^{\ell+1} C_i$, which costs more than B, must have an isolation weight of at least $\ell/2$, implying that the heavier cut between $\bigcup_{i=1}^m C_i$ and C_{m+1} has weight at least $\ell/4$. □

3.3.2. A $(\frac{1}{3} - \epsilon)$-approximation

It can be readily seen from the analysis of Algorithm 2 that improving the approximation factor requires an efficient use of the given budget. To this end, we use as a procedure the FPTAS for the *knapsack* problem presented in [23], denoted by $A(\pi, \epsilon)$, where π is the knapsack instance.

1: **for** each $s \in S$ **do**
2: Find a minimum cost isolating cut for s, and denote it by C_s.
3: **end for**
4: Sort the cuts satisfying $c(C_s) \leqslant B$, $s \in S$, in non-decreasing order of the ratio between their cost and the weight of their terminal $(c(C_s)/w(s))$. Let $\{C_i\}_{i=1}^k$ be the resulting sequence of cuts.
5: Let $\{C_i\}_{i=1}^m$ be the maximal prefix of $\{C_i\}_{i=1}^k$ with a total cost of at most B. Choose the heavier cut (with respect to isolation weight) between $\bigcup_{i=1}^m C_i$ and C_{m+1} (if $m = k$ then $\bigcup_{i=1}^m C_i$ is an optimal solution).

Algorithm 2. A greedy algorithm for weighted BIMC.

1: **for** each $s \in S$ **do**
2: Find a minimum cost isolating cut for s, and denote it by C_s.
3: **end for**
4: Construct an instance of the *knapsack* problem, π, as follows: treat each terminal $s \in S$ such that $c(C_s) \leqslant B$ as an item whose profit is $w(s)$ and whose size is $c(C_s)$, and let B be the capacity of the knapsack.
5: Run $A(\pi, \epsilon)$ and denote by P the resulting subset of terminals. Output the cut $\bigcup_{s \in P} C_s$.

Algorithm 3. A $(\frac{1}{3} - \epsilon)$-approximation for weighted BSMC.

Let OPT be an optimal solution for the weighted BIMC instance. Since every terminal s, for which $c(C_s) > B$, cannot be isolated by either OPT or Algorithm 3, we ignore such terminals in what follows. Let I denote the set of the terminals isolated by OPT and let ℓ be the isolation weight of OPT, i.e., the value of the optimal solution. Denote by $|OPT(\pi)|$ the value of the optimal solution for the *knapsack* instance π.

Lemma 20. $|OPT(\pi)| \geqslant \frac{1}{3}\ell$.

Proof. Let $U = \{X \subseteq I \mid \sum_{s \in X} w(s) \geqslant \frac{1}{3}\ell\}$, i.e., U is the collection of subsets of I having profit at least $\frac{1}{3}\ell$. Let $Y \in U$ be a subset of minimum *size in* π (notice that there must exist such a subset). Assume to the contrary that $|OPT(\pi)| < \frac{1}{3}\ell$, and in particular that $\sum_{s \in Y} c(C_s) > B$. As every single item is a feasible solution by itself, it follows that for every item s, $w(s) < \frac{1}{3}\ell$. Thus, there are at least two terminals in Y, and moreover, $\sum_{s \in Y} w(s) < \frac{2}{3}\ell$ (otherwise, by taking off a terminal from Y we get a contradiction to the minimality of Y in U with respect to size). By arguments similar to those used in the proof of Lemma 17, we get that $\sum_{s \in I} c(C_s) \leqslant 2B$. Thus,

$$\sum_{s \in I \setminus Y} c(C_s) < B,$$

and

$$\sum_{s \in I \setminus Y} w(s) > \frac{1}{3}\ell.$$

Thus, $I \setminus Y$ is a feasible solution to π with the desired value. □

Lemma 21. *Algorithm 3 achieves an approximation factor of* $\frac{1}{3} - \epsilon$.

Proof. Follows from Lemma 20 and the FPTAS for the *knapsack* problem. □

We note that it can be shown that Lemma 20 is tight for arbitrarily large values of k by constructing appropriate examples.

In what follows we show that given a lower bound on the optimal solution's value (for example, the value of the solution returned by Algorithm 3), the analysis performed in Lemma 20 can be improved for some instances. First, we generalize the definition of the sets U and Y as follows: for $i > 1$, let

$$U_i = \left\{ X \subseteq I \,\middle|\, \sum_{s \in X} w(s) \geqslant \left(\frac{1}{2} - \frac{1}{4i-2}\right)\ell \right\}$$

and let Y_i be a set in U_i of minimum *size in* π. Notice that $U = U_2$ and $Y = Y_2$. The following lemma, generalizes Lemma 20.

Lemma 22. *If, for every* $X \in U_i$, $|X| \geqslant i$, *then* $|OPT(\pi)| \geqslant (\frac{1}{2} - \frac{1}{4i-2})\ell$.

Proof. Assume by contradiction that $|OPT(\pi)| < (\frac{1}{2} - \frac{1}{4i-2})\ell$ and in particular that $\sum_{s \in Y_i} c(C_s) > B$. Recall that $|Y_i| \geqslant i$ and that any subset of Y_i of size $i - 1$ must have a total weight less than $(\frac{1}{2} - \frac{1}{4i-2})\ell$. Thus, there must exist

a terminal in Y_i that has a weight of at most

$$\left(\frac{1}{2} - \frac{1}{4i-2}\right) \cdot \frac{\ell}{i-1} \leqslant \frac{\ell}{2i-1},$$

and thus, from the minimality of Y_i in U with respect to size,

$$\sum_{s \in Y_i} w(s) < \left(\frac{1}{2} + \frac{1}{4i-2}\right)\ell.$$

By similar arguments to those used in the proof of Lemma 20, we get that $\sum_{s \in I \setminus Y_i} c(C_s) < B$ and $\sum_{s \in I \setminus Y_i} w(s) > (\frac{1}{2} - \frac{1}{4i-2})\ell$ and thus $I \setminus Y_i$ is a feasible solution to π with the desired value. □

Now, given an instance of weighted BIMC, let ℓ' be a lower bound on l. Define

$$U_i' = \left\{ X \subseteq S \,\middle|\, \sum_{s \in X} w(s) \geqslant \left(\frac{1}{2} - \frac{1}{4i-2}\right)\ell' \right\}.$$

Since $U_i \subseteq U_i'$, if for every $X \in U_i'$ it holds that $|X| \geqslant i$, it follows from Lemma 22 that the solution returned by Algorithm 3 is within $\frac{1}{2} - \frac{1}{4i-2} - \epsilon$ of the optimal solution. Notice that finding such maximal i can be done efficiently.

4. The weighted budgeted separating multiway cut problem

In this section, we study the weighted BSMC problem which is equivalent to the weighted budgeted variant of the multicut problem. We show that approximating it is at least as hard as the sparsest cut problem. We present a natural linear programming relaxation for the problem and show that it has an unbounded integrality gap for general graphs. We notice that the weighted BSMC on *trees* is a special case of the *maximum coverage problem*, and hence it can be approximated using the algorithm of Khuller, Moss and Naor [11]. Moreover, we prove that their algorithm's output is within a constant factor from the optimal *fractional* solution of the aforementioned linear relaxation, implying a constant integrality gap for tree instances. Lastly, we use a recent hierarchical decomposition of graphs by Räcke (see [12] and [13]) to obtain a bi-criteria approximation of $(\frac{e}{e-1}, O(\log^2 n \log\log n))$ for arbitrary graphs.

4.1. Hardness results

4.1.1. Hardness with respect to the sparsest cut problem

We first prove the following lemma.

Lemma 23. *Given a non-empty cut $C \subseteq E$ that partitions G into $r > 2$ connected components, there is an algorithm that finds a cut $C' \subset C$ such that $c(C')/w(C') \leqslant c(C)/w(C)$ and C' partitions G into $r-1$ connected components.*

Proof. Given the r connected components into which G is partitioned by C, denote by V_i the vertex set of the ith connected component. Define:

$$C_{ij} = \{e = (u, v) \in C\colon u \in V_i \wedge v \in V_j\}.$$

Obviously, $C = \bigcup_{1 \leqslant i < j \leqslant r} C_{ij}$ and $C_{ij} \cap C_{\ell m} = \emptyset$ for every $(i, j) \neq (\ell, m)$. Define the separation weight of C_{ij}, denoted by $w(C_{ij})$, as the sum of the weights of the pairs of terminals (s_g, s_h) such that $s_g \in S \cap V_i$ and $s_h \in S \cap V_j$. Then:

$$w(C) \geqslant \sum_{1 \leqslant i < j \leqslant r} w(C_{ij}) \quad \text{and} \quad c(C) = \sum_{1 \leqslant i < j \leqslant r} c(C_{ij}). \tag{1}$$

Proposition 24. *There exists a C_{ij} $(1 \leqslant i < j \leqslant r)$ such that $c(C_{ij})/w(C_{ij}) \geqslant c(C)/w(C)$.*

Proof. Assume to the contrary that for all $1 \leqslant i < j \leqslant r$, $c(C_{ij})/w(C_{ij}) < c(C)/w(C)$. Thus, $c(C_{ij}) \cdot w(C) < c(C) \cdot w(C_{ij})$, and by summing up over all $1 \leqslant i < j \leqslant r$ and from Equality (1), we get that

$$c(C) \cdot w(C) = \sum_{1 \leqslant i < j \leqslant r} c(C_{ij}) \cdot w(C) < c(C) \cdot \sum_{1 \leqslant i < j \leqslant r} w(C_{ij})$$

and thus $w(C) < \sum_{1 \leqslant i < j \leqslant r} w(C_{ij})$ contradicting inequality (1). □

Note that a C_{ij} $(1 \leqslant i < j \leqslant r)$ satisfying the above proposition can be found easily. Now, define $C' = C \setminus C_{ij}$, and observe that C' partitions G into $r - 1$ connected components. (The previous connected components of V_i and V_j are merged into one, and the rest are not changed.) Now, since $c(C_{ij})/w(C_{ij}) \geqslant c(C)/w(C)$, we get:

$$c(C_{ij}) \cdot w(C) \geqslant c(C) \cdot w(C_{ij})$$

and

$$\big(c(C) - c(C_{ij})\big) \cdot w(C) \leqslant c(C) \cdot \big(w(C) - w(C_{ij})\big).$$

Thus,

$$c(C')/w(C') = \big(c(C) - c(C_{ij})\big)/\big(w(C) - w(C_{ij})\big) \leqslant c(C)/w(C)$$

and the lemma follows. □

Corollary 25. *Given a non-empty cut $C \subseteq E$, there is an algorithm that finds a non-empty subset of vertices $U \subseteq V$ such that the sparsity of the cut associated with U is at most $c(C)/w(C)$.*

Proof. Assume that cut C partitions G into $r > 1$ connected components, and denote by V_i the vertex set of the ith connected component. If $r = 2$, then V_1 can be returned. Otherwise, apply recursively the algorithm from Lemma 23 until a cut C' that partitions G into two connected components is obtained. Let V_1' be the vertices of one of these connected components. Then, V_1' can be returned. □

The following theorem shows that the weighted BSMC problem is at least as hard to approximate as the sparsest cut problem is (up to a constant).

Theorem 26. *Let ALG be an (α, β)-approximation for weighted BSMC. Then, there exists a $(1+\epsilon)\alpha\beta$-approximation for Sparsest Cut, for every $\epsilon > 0$.*

Proof. Assume we are given an instance of the sparsest cut problem, denote it by π and let OPT_π denote its optimal solution. Denote the sparsity of the optimal solution by

$$|OPT_\pi| = \frac{c(OPT_\pi, \overline{OPT_\pi})}{w(OPT_\pi, \overline{OPT_\pi})}.$$

Let (π, B) denote the input for the weighted BSMC problem that consists of the instance π and the budget B, and let $OPT_{\pi,B}$ be a corresponding optimal solution. Then, since $(OPT_\pi, \overline{OPT_\pi})$ is a feasible solution for the weighted BSMC problem on (π, B) for every $B \geqslant c(OPT_\pi, \overline{OPT_\pi})$, then $w(OPT_\pi, \overline{OPT_\pi}) \leqslant w(OPT_{\pi,B})$ for every $B \geqslant c(OPT_\pi, \overline{OPT_\pi})$.

For $\lceil \log_{1+\epsilon} c(C_{\min}) \rceil \leqslant i \leqslant \lceil \log_{1+\epsilon} c(E) \rceil$, where $C_{\min}$ is the minimum cost cut in G, let C_{B_i} be the cut returned by $ALG(\pi, B_i = (1+\epsilon)^i)$. Then, by applying Corollary 25 on each C_{B_i} we can obtain a non-empty subset of vertices $U_i \subseteq V$ such that the sparsity of the cut associated with U_i is at most

$$\frac{c(C_{B_i})}{w(C_{B_i})} \leqslant \frac{\beta B_i}{w(OPT_{\pi,B_i})/\alpha} = \alpha\beta \frac{B_i}{w(OPT_{\pi,B_i})}.$$

Let $j = \lceil \log_{1+\epsilon} c(OPT_\pi, \overline{OPT_\pi}) \rceil$. Then,

$$\frac{B_j}{w(OPT_{\pi,B_j})} \leqslant (1+\epsilon)\frac{c(OPT_\pi, \overline{OPT_\pi})}{w(OPT_\pi, \overline{OPT_\pi})} = (1+\epsilon)|OPT_\pi|.$$

We conclude that the sparsity of $(U_j, \overline{U_j})$ is at most $(1+\epsilon)\alpha\beta|OPT_\pi|$, and the theorem follows by outputting the sparsest cut among the computed cuts

$$\left\{(U_i, \overline{U_i})\right\}_{\lceil \log_{1+\epsilon} c(C_{\min})\rceil \leqslant i \leqslant \lceil \log_{1+\epsilon} c(E)\rceil}. \qquad \square$$

4.1.2. Integrality gap of a linear programming relaxation

In this subsection we give a natural linear programming relaxation for the weighted BSMC problem. We assign an indicator variable y_e for every edge $e \in E$ (we assume without loss of generality that $c(e) \leqslant B$ for every $e \in E$). The budget constraint can be stated accordingly. We assume that $S = \{s_1, \ldots, s_k\}$ and assign an indicator variable x_{ij} for every pair of terminals $s_i, s_j \in S$ indicating whether the pair (s_i, s_j) is separated by the given solution. The separation constraint of a pair of terminals (s_i, s_j) is $x_{ij} \leqslant \sum_{e \in P_{i,j}} y_e$, for each path $P_{i,j}$ between s_i and s_j.

(SEP-LP)

$$\begin{aligned}
&\max \ \textstyle\sum_{s_i, s_j \in S} w(s_i, s_j) \cdot x_{ij} \\
&\text{s.t.} \\
&\quad x_{ij} - \textstyle\sum_{e \in P_{i,j}} y_e \leqslant 0 && \text{for every} s_i, s_j \in S \\
&\quad && \text{and path } P_{i,j} \text{ from } s_i \text{ to } s_j \\
&\quad \textstyle\sum_{e \in E} c(e) \cdot y_e \leqslant B \\
&\quad 0 \leqslant x_{ij} \leqslant 1 && \text{for every } s_i, s_j \in S \\
&\quad 0 \leqslant y_e && \text{for every } e \in E
\end{aligned}$$

Proposition 27. *The integrality gap of SEP-LP is* $\Omega(n)$.

Proof. Consider the following graph that consists of 3 parts:

- A square of non-terminals $\{v_1, v_2, v_3, v_4\}$ and edges with cost of $c(e) = 1 + \epsilon$.
- $N/2$ terminals are connected to v_1 by edges with $c(e) = 2$.
- $N/2$ terminals are connected to v_3 by edges with $c(e) = 2$.

Let $B = 2$, and $w(s_i, s_j) = 1$ for every $s_i, s_j \in S$. An optimal solution picks any of the non-square edges and has a value of $N - 1$, while a feasible optimal fractional solution is: $y_e = 1/(1+\epsilon)$ for the square edges (v_2, v_3) and (v_3, v_4), and $x_{ij} = 1/(1+\epsilon)$ for every pair of terminals (s_i, s_j) such that $(s_i, v_1) \in E$ and $(s_j, v_3) \in E$. This solution has a value of $\frac{N^2}{4(1+\epsilon)}$. $\square$

This proposition implies that an algorithm based on the above linear relaxation would have poor performance. Nevertheless, in what follows we show an approximation algorithm for the special case of trees based on this linear relaxation.

4.2. Approximation algorithms for weighted BSMC in trees

As the multicut problem on trees is *NP*-hard, so is the BSMC problem on trees. At the same time, like there are better approximation algorithms for the multicut problem on trees than for multicut on general instances, BSMC is apparently easier when restricted to trees, as it is a special case of the maximum coverage problem. The elements are the pairs of terminals, and the edges are the sets. The set that corresponds to an edge e contains a pair of terminals (s_i, s_j) if and only if e belongs to the unique path between s_i and s_j. The weight of an element is the weight of the corresponding pair, while the cost of a set is the cost of the corresponding edge.

In [11], Khuller, Moss and Naor presented an $\frac{e-1}{e}$-approximation algorithm for the maximum coverage problem, and accordingly, the same algorithm can be used to approximate BSMC on trees. We note that the analysis of [11] im-

plies that the algorithm's output is within a factor of $\frac{e-1}{e}$ from the optimal *integral* solution, but presents no guarantee on the ratio between the output and the optimal *fractional* solution.

In what follows we provide a dual-fitting analysis for the same algorithm, which proves that the algorithm's output is within a factor of $\frac{1}{3}$ from the optimal *fractional* solution (we note that our analysis holds for a general instance of the maximum coverage problem). This type of guarantee might be important for some applications (for example, see [24]).

The weighted BSMC problem on trees can be cast as a linear integer program, whose fractional relaxation is SEP-LP. Notice that there is a unique path in the given tree between s_i and s_j, denoted by P_{ij}. The dual LP of SEP-LP is:

$$\begin{aligned}
&\min\ B\cdot\gamma+\textstyle\sum_{s_i,s_j\in S}\beta_{ij} && \text{(SEP-DLP)}\\
&\text{s.t.}\\
&\quad c(e)\cdot\gamma-\textstyle\sum_{i,j:\ e\in P_{ij}}\alpha_{ij}\geqslant 0 && \text{for every } e\in E\\
&\quad \alpha_{ij}+\beta_{ij}\geqslant w(s_i,s_j) && \text{for every } s_i,s_j\in S\\
&\quad \alpha_{ij}\geqslant 0 && \text{for every } s_i,s_j\in S\\
&\quad \beta_{ij}\geqslant 0 && \text{for every } s_i,s_j\in S\\
&\quad \gamma\geqslant 0
\end{aligned}$$

We define the *worthiness of an edge e with respect to C*, a feasible solution, as

$$\Gamma_C(e)=\frac{\sum_{i,j:\ e\in P_{ij}} w(s_i,s_j)\cdot(1-x_{ij})}{c(e)},$$

where x is the corresponding solution of SEP-LP. Algorithm 4 greedily adds edges to the solution as long as the budget constraint is not violated. At the same time it maintains the corresponding solution of SEP-LP. Note that we may assume without loss of generality that for all $e\in E$, $c(e)\leqslant B$.

Observation 28. *If $C\neq E$, then $\sum_{h=0}^{|C|-1} c(e_h)+c(e_{|C|})>B$.*

The following proposition follows immediately from the definition of the worthiness of an edge and the fact that edges are only added to the solution during the algorithm.

Proposition 29. *For every $e\in E$ and $0<h\leqslant|C|$, $\Gamma_{C_h}(e)\leqslant\Gamma_{C_{h-1}}(e)$, i.e., the worthiness of an edge can only decrease during the algorithm.*

Corollary 30. *If $C\neq E$, then*

$$c(e_{|C|})\cdot\Gamma_C(e_{|C|})\leqslant c(e_{|C|})\cdot\Gamma_{C_0}(e_{|C|})=w\big(\{e_{|C|}\}\big),$$

i.e., adding the edge $e_{|C|}$ to C increases its separation weight by at most the separation weight of $\{e_{|C|}\}$.

Corollary 31. *For every $0<h\leqslant|C|$, $\Gamma_{C_h}(e_h)\leqslant\Gamma_{C_{h-1}}(e_{h-1})$.*

Proof. From Proposition 29, $\Gamma_{C_h}(e_h)\leqslant\Gamma_{C_{h-1}}(e_h)$, and the corollary follows from the greediness of Algorithm 4. □

1: Initialize: $h=0$, $C_0=\emptyset$, $x_{ij}=0$ for every $s_i,s_j\in S$ and $y_e=0$ for every $e\in E$.
2: **while** $\exists e\in E\setminus C_h$ **do**
3: Let $e_h\in E\setminus C_h$ be an edge with the lowest cost among the edges with the maximum value of Γ_{C_h}.
4: If $c(e_h)>B-c(C_h)$, output the better solution between $\{e_h\}$ and $C=C_h$.
5: $C_{h+1}\leftarrow C_h\cup\{e_h\}$.
6: Set $y_{e_h}=1$ and $x_{ij}=1$ for all the pairs of terminals (s_i,s_j) separated by e_h.
7: $h\leftarrow h+1$
8: **end while**
9: Output $C=C_h$.

Algorithm 4. A greedy algorithm for weighted BSMC on trees.

Observation 32. $w(C) = \sum_{h=0}^{|C|-1} c(e_h) \cdot \Gamma_{C_h}(e_h)$.

Theorem 33. *Algorithm 4 returns a solution which is within a factor of* $\frac{1}{3}$ *from the optimal* fractional *solution of SEP-LP.*

Proof. If the algorithm reached Step 9, then $C = E$, and the solution is optimal. Otherwise, denote by $w(ALG)$ the value of the solution output by Algorithm 4. Consider the following dual solution:

$$\beta_{ij} = w(s_i, s_j) \cdot x_{ij}, \qquad \alpha_{ij} = w(s_i, s_j) \cdot (1 - x_{ij}), \qquad \gamma = \Gamma_C(e_{|C|}).$$

Since $\Gamma_C(e_{|C|}) \geqslant \Gamma_C(e)$ for every $e \notin C$, this is a feasible dual solution. Let z denote its value. Then,

$$z = B \cdot \gamma + \sum_{s_i, s_j \in S} \beta_{ij} \tag{2}$$

$$= B \cdot \Gamma_C(e_{|C|}) + \sum_{s_i, s_j \in S} w(s_i, s_j) \cdot x_{ij} \tag{3}$$

$$< \left(\sum_{h=0}^{|C|-1} c(e_h) + c(e_{|C|}) \right) \cdot \Gamma_C(e_{|C|}) + w(C) \tag{4}$$

$$\leqslant \sum_{h=0}^{|C|-1} c(e_h) \cdot \Gamma_C(e_{|C|}) + w\big(\{e_{|C|}\}\big) + w(C) \tag{5}$$

$$\leqslant \sum_{h=0}^{|C|-1} c(e_h) \cdot \Gamma_{C_h}(e_h) + w\big(\{e_{|C|}\}\big) + w(C) \tag{6}$$

$$= w(C) + w\big(\{e_{|C|}\}\big) + w(C) \tag{7}$$

$$\leqslant 3w(ALG), \tag{8}$$

where inequality (4) follows from the definition of the Algorithm 4 and Observation 28, (5) follows from Corollary 30, (6) follows from Corollary 31 and (7) follows from Observation 32. Thus, the theorem follows by weak duality. □

4.3. An approximation algorithm for weighted BSMC on general graphs

In this subsection we present an $(\frac{e}{e-1}, O(\log^2 n \log\log n))$-approximation algorithm for weighted BSMC.

In [12], Räcke describes a hierarchical decomposition of any undirected graph $G = (V, E)$ into a tree T_G, where there is a 1–1 correspondence between V and the leaves of T_G. T_G has the property that any feasible multi-commodity flow function in T_G can be routed in G causing a congestion bounded by a function of G's parameters, denoted by β. By min-cut-max-flow theorems this implies a corresponding bounded ratio between the cost of cuts in G and the cost of cuts in T_G. In [13], Harrelson, Hildrum and Rao give a polynomial-time construction of T_G with $\beta = O(\log^2 n \log\log n)$, which we use in the following algorithm.

Theorem 34. *Algorithm 5 is a* $(\frac{e}{e-1}, O(\log^2 n \log\log n))$*-approximation for the weighted BSMC problem.*

1: Let $B' = 2\beta B$.
2: Construct a decomposition tree, T_G, of G.
3: **for** $\forall e = (u, v) \in T_G$ with a cost $> B'$ **do**
4: Merge the vertices u and v.
5: **end for**
6: Let T'_G be the resulted tree.
7: Run Algorithm 4 on T'_G with budget B', and output the associated cut in G.

Algorithm 5. A bi-criteria approximation algorithm for weighted BSMC.

1: Compute a Gomory–Hu tree T for G.
2: Sort the edges of T in a non-decreasing order of their cost.
3: Choose the maximal sequence of edges starting from the cheapest, whose cost is at most B, and output the union of the cuts associated with these edges in G, denoted by C.

Algorithm 6. A greedy algorithm for BGD.

Proof. Let OPT be an optimal solution, and let I denote the set of pairs of terminals separated by OPT. Let OPT_{T_G} be a minimum cost cut separating I in T_G. By [25], $c(OPT_{T_G}) \leqslant 2MCF_I(T_G)$, where $MCF_I(T_G)$ is the value of the maximum multi-commodity flow in T_G between the pairs in I. By the construction of T_G and its property, $MCF_I(T_G) \leqslant \beta MCF_I(G)$. Since $MCF_I(G)$ lower bounds the cost of any cut separating I in G, $MCF_I(G) \leqslant c(OPT)$, and thus we get

$$c(OPT_{T_G}) \leqslant 2\beta c(OPT) \leqslant 2\beta B = B'.$$

In particular, OPT_{T_G} does not contain any edge with cost more than B', and thus OPT_{T_G} is a feasible solution for the weighted BSMC problem on T'_G with budget B', with value $w(OPT_{T_G}) \geqslant w(OPT)$. From [11], running Algorithm 4 will return a solution C whose cost is at most B' and whose value is at least $\frac{e-1}{e} w(OPT_{T_G})$. By the properties of the decomposition tree, the associated cut in G has a cost of at most B' and a separation weight of at least $w(C)$ and the theorem follows. □

Since the weighted budgeted variant of Multicut is equivalent to weighted BSMC, we conclude that a bi-criteria $(\frac{e}{e-1}, O(\log^2 n \log\log n))$-approximation exists for this problem as well.

5. The budgeted graph disconnection problem

The following algorithm for BGD is a variant of the algorithm presented in [15] for the k-cut problem. In what follows, we refer to the algorithm for the k-cut problem and its proof as they appear in [26, pp. 40–44].

Lemma 35. *Let ℓ denote the value of an optimal solution. Algorithm 6 achieves an approximation factor of $\frac{1}{2} + \frac{1}{\ell}$ if ℓ is even, and $\frac{1}{2} + \frac{1}{2\ell}$ if ℓ is odd.*

Proof. The algorithm for the k-cut problem [26] outputs the union of the lightest $k-1$ cuts of the cuts associated with edges of T in G and achieves an approximation factor of $2 - 2/k$. In particular, the cost of the lightest $k-1$ edges of T is at most $(2-2/k)|OPT_k|$, where $|OPT_k|$ is the cost of an optimal k-cut. Assume without loss of generality that OPT is an optimal ℓ-cut. Then, the cost of the lightest $\ell - 1$ edges of T is at most $(2-2/\ell)B$, and thus the cost of the lightest $\lfloor(\ell-1)/(2-2/\ell)\rfloor = \lfloor \ell/2 \rfloor$ is at most B. Thus, C is associated with at least $\lfloor \ell/2 \rfloor$ edges, and partitions G to at least $\lfloor \ell/2 \rfloor + 1$ connected components. Noticing that the feasibility of C follows from the properties of Gomory–Hu trees completes the proof. □

We note that Example 4.9 in [26] shows that the above analysis is tight.

6. Further discussion

Among the problems that were studied in this paper, the weighted BSMC problem seems the hardest. It remains an open question whether it is possible to improve upon the bi-criteria approximation that we presented or even achieve a uni-criteria approximation. In this section we review some related ideas and point out some possible directions towards solving the problem.

6.1. The budgeted sparsest cut problem

Consider the following budget problem, whose input is the same as the input for the weighted BSMC problem.

Problem 36 *(Budgeted Sparsest Cut).* Find a non-empty subset of vertices $U \subset V$ such that $c(U, \overline{U}) \leqslant B$ and the sparsity of $(U, \overline{U})$ is minimized.

In order to understand the relation between weighted BSMC and Budgeted Sparsest Cut, we look for results similar to those of Section 4.1.1. Notice that the algorithm of Corollary 25 actually finds a cut whose cost is at most $c(C)$. Hence, Theorem 26 can be easily generalized to obtain the following.

Theorem 37. *Let ALG be an (α, β)-approximation for weighted BSMC. Then, there exists a $((1+\epsilon)\alpha\beta, \beta)$-approximation for the Budgeted Sparsest Cut problem for every $\epsilon > 0$.*

Specifically, notice that a uni-criteria approximation for weighted BSMC implies an appropriate uni-criteria approximation for the Budgeted Sparsest Cut problem.

6.2. Linear programming formulation revisited

In Section 4.1.2 we introduced SEP-LP, a linear programming relaxation for weighted BSMC. As its integrality gap is $\Omega(n)$, using SEP-LP to obtain a good approximation seems unlikely. Nevertheless, one might look for a different linear programming formulation with a smaller integrality gap. To this end, reconsider the example that was used to prove the integrality gap of SEP-LP in Proposition 27. Notice that the optimal fractional solution fractionally buys a cut whose cost is more than the given budget. Accordingly, adding a constraint to forbid solutions that buy expensive cuts might improve the linear programming formulation.

We consider the following modified version of SEP-LP, in which the variables x_{ij} and y_e have the same meaning, and we only replace the separation constraints ($x_{ij} \leqslant \sum_{e \in P_{i,j}} y_e$). Let $L = \{U \subseteq V \mid c(U, \overline{U}) \leqslant B\}$, i.e., $U \in L$ if the cost of its associated cut is at most B. We assign a variable z_U for every cut $U \in L$. In what follows we show that the set of integer feasible solutions to SEP-LP2 corresponds to the set of the feasible cuts for the BSMC problem.

$$
\begin{array}{lll}
\max \ \sum_{s_i, s_j \in S} w(s_i, s_j) \cdot x_{ij} & & \text{(SEP-LP2)} \\
\text{s.t.} & & \\
\quad x_{ij} - \frac{1}{2} \sum_{U \in L:\ |U \cap \{s_i, s_j\}| = 1} z_U \leqslant 0 & \text{for every } s_i, s_j \in S & \\
\quad \frac{1}{2} \sum_{U \in L:\ |U \cap e| = 1} z_U - y_e \leqslant 0 & \text{for every } e \in E & \\
\quad \sum_{e \in E} c(e) \cdot y_e \leqslant B & & \\
\quad 0 \leqslant x_{ij} \leqslant 1 & \text{for every } s_i, s_j \in S & \\
\quad 0 \leqslant y_e & \text{for every } e \in E & \\
\quad 0 \leqslant z_U & \text{for every } U \in L &
\end{array}
$$

Let C be a feasible solution to a given BSMC instance. Denote by $U_1, \ldots, U_k$ the set of connected components into which the graph is partitioned by C. Note that by feasibility of C, $U_i \in L$ for $1 \leqslant i \leqslant k$. Set: $y_e = 1$ for every $e \in C$ and $y_e = 0$ otherwise; $x_{ij} = 1$ for every i, j such that C separates s_i and s_j and $x_{ij} = 0$ otherwise; $z_U = 1$ for every $U \in \{U_i\}_{i=1}^k$ and $z_U = 0$ otherwise. Consider a pair of terminals separated by C, s_i and s_j, and assume that $s_i \in U_i$ and $s_j \in U_j$. Then, since $U_i \neq U_j$ we have that

$$x_{ij} = 1 = \frac{1}{2} \cdot 2 = \frac{1}{2} \cdot (z_{U_i} + z_{U_j}) \leqslant \frac{1}{2} \sum_{U \in L:\ |U \cap \{s_i, s_j\}| = 1} z_U.$$

Next, consider an edge $e \in E$. If $e \subseteq U_i$ for some $1 \leqslant i \leqslant k$, we have that $\sum_{U \in L:\ |U \cap e| = 1} z_U = 0$. Otherwise, e has one endpoint in U_i and one endpoint in U_j, and specifically, $e \in C$. Hence, we have that

$$\frac{1}{2} \sum_{U \in L:\ |U \cap e| = 1} z_U = \frac{1}{2} \cdot (z_{U_i} + z_{U_j}) = 1 = y_e.$$

We conclude that the above solution is a feasible integer solution for SEP-LP2 with the same value as C.

Next, given $\{x_{ij}\}_{s_i, s_j \in S}$, $\{y_e\}_{e \in E}$, $\{z_U\}_{U \in L}$, an integer feasible solution to SEP-LP2, consider the cut

$$C = \bigcup_{z_{U_i} \geqslant 1} (U_i, \overline{U_i}).$$

Let $e \in C$. It follows by the definition of C that there exists U_i such that $z_{U_i} \geqslant 1$ and $|U_i \cap e| = 1$. Hence, by feasibility we have:

$$\frac{1}{2} \leqslant \frac{1}{2} \cdot z_{U_i} \leqslant \frac{1}{2} \sum_{U \in L:\ |U \cap e| = 1} z_U \leqslant y_e,$$

and thus, by integrality, $y_e \geqslant 1$. Accordingly, by feasibility we have that:

$$\sum_{e \in C} c(e) \leqslant \sum_{e \in C} c(e) \cdot y_e \leqslant B,$$

which means that C is a feasible solution. Consider a pair of terminals, s_i and s_j, such that $x_{ij} = 1$. By feasibility and integrality, there exists U_i such that $z_{U_i} \geqslant 1$ and $(U_i, \overline{U_i})$ separates s_i and s_j. It follows by the definition of C that s_i and s_j are also separated by C. We conclude that C is a feasible solution whose value is at least the value of the given solution for SEP-LP2.

As the size of SEP-LP2 is exponential, the dual program should be considered.

$$\begin{array}{lll}
\min\ B \cdot \gamma + \sum_{s_i, s_j \in S} \delta_{ij} & & \text{(SEP-DLP2)} \\
\text{s.t.} & & \\
\quad \alpha_{ij} + \delta_{ij} \geqslant w(s_i, s_j) & \text{for every } s_i, s_j \in S & \\
\quad c(e) \cdot \gamma - \beta_e \geqslant 0 & \text{for every } e \in E & \\
\quad \sum_{|U \cap e| = 1} \beta_e - \sum_{|U \cap \{s_i, s_j\}| = 1} \alpha_{ij} \geqslant 0 & \text{for every } U \in L & \\
\quad \alpha_{ij} \geqslant 0 & \text{for every } s_i, s_j \in S & \\
\quad \beta_e \geqslant 0 & \text{for every } e \in E & \\
\quad \gamma \geqslant 0 & & \\
\quad \delta_{ij} \geqslant 0 & \text{for every } s_i, s_j \in S &
\end{array}$$

In order to solve SEP-DLP2 a separation oracle is needed, as the number of constraints might be exponential. However, observe that such an oracle should find a cut whose sparsity (with respect to α_{ij} as pair weights and β_e as edge costs) is less than 1, among cuts belonging to L. This problem generalizes the budgeted sparsest cut problem, as it deals with finding a sparsest cut among a given set of cuts (here, the interesting cuts are not necessarily the cheap cuts with respect to the edge costs). It is reasonable to look for algorithms for weighted BSMC that use an approximation algorithm for this problem to obtain an approximate solution to SEP-DLP2 (and an appropriate solution to SEP-LP2). However, we did not succeed to prove a general theorem, analogous to Theorem 37, but only a result for a special case, which is presented in the following subsection.

6.3. Node-weighted BSMC

The node-weighted BSMC problem is a special case of the weighted BSMC problem in which $w(s_i, s_j) = \varphi(s_i) \cdot \varphi(s_j)$, where $\varphi : V \to \mathbb{R}^+$ is a given node weight function. In what follows we assume without loss of generality that φ is normalized so that the minimal positive weight of a node is exactly 1.

Theorem 38. *Let ALG be an (α, β)-approximation for the budgeted sparsest cut problem. Then, there exists an $(O(\alpha), \beta + 1)$-approximation for the node-weighted BSMC problem.*

Proof. Let $\Phi = \sum_{s_i \in S} \varphi(s_i)$, and notice that $\sum_{s_i \neq s_j} w(s_i, s_j) \leqslant \Phi^2/2$, hence $\Phi^2/2$ is an upper bound on the value of the optimal solution. We assume without loss of generality that for every pair (s_i, s_j), there exists a separating cut whose cost is at most B. If that is not the case, such a pair of terminals can be merged into a new terminal s' with $\varphi(s') = \varphi(s_i) + \varphi(s_j)$, as no feasible solution can separate such pair.

Consider the following iterative algorithm. In each iteration we run ALG to find a "good" cut in some connected component and add its edges to the solution. If the solution's cost exceeds the budget, we terminate with the current solution. In the first iteration, the input for ALG is the given graph G, and the given budget B. The output of ALG in the ith iteration is a cut C_i that separates its input to two connected components. The connected component with the bigger node weight is the input for the next iteration, along with the budget B. Notice that the algorithm may also

terminate if the current connected component contains only one vertex with a positive weight. Let C be the output of the algorithm.

Obviously, the above algorithm runs in polynomial time, as the number of iteration is bounded by the number of edges in the graph. Furthermore, by the termination criteria and the input budget for ALG, $c(C) \leqslant (\beta+1)\cdot B$. Let OPT denote the optimal solution, and denote its separation weight by $w(OPT) = z\cdot\Phi^2$. Notice that as aforesaid $z \leqslant 1/2$. We also have that

$$\forall s_i \in S, \quad w(OPT) \leqslant \Phi\cdot\big(\Phi-\varphi(s_i)\big), \tag{9}$$

as the optimal solution cannot do better than separating all the terminals. In what follows we prove that $w(C) = \frac{w(OPT)}{\mathrm{O}(\alpha)}$.

Let h denote the total number of iterations, and $\gamma_i\cdot\Phi$ be the node weight of the connected component with the *smaller* node weight among the components that were separated during the ith iteration. Similarly, let $\gamma_{h+1}\cdot\Phi$ be the node weight of the connected component with the *bigger* node weight among the components that were separated during the last iteration. Note that for $i \leqslant h$, $\sum_{j=i+1}^{h+1}\gamma_j \geqslant \gamma_i$ as among the two components that were separated during the ith iteration, $\sum_{j=i+1}^{h+1}\gamma_j$ is the node weight of the bigger component while γ_i is the node weight of the smaller one.

If for some $i \leqslant h$, $\gamma_i \geqslant 1/4$, then since $\sum_{j=i+1}^{h+1}\gamma_j \geqslant \gamma_i$, we have that $w(C) \geqslant \frac{\Phi^2}{16} \geqslant \frac{w(OPT)}{8}$ and the theorem follows. Otherwise, if $\sum_{i=1}^{h}\gamma_i > \frac{z}{2}$, then

$$w(C) = \frac{\Phi^2}{2}\cdot\sum_{i=1}^{h+1}\gamma_i\cdot(1-\gamma_i) \geqslant \frac{\Phi^2}{2}\cdot\sum_{i=1}^{h}\gamma_i\cdot\frac{3}{4} \geqslant \frac{3\Phi^2\cdot z}{16} = \frac{w(OPT)}{16/3},$$

and the theorem follows.

Now, assume that $\sum_{i=1}^{h}\gamma_i \leqslant \frac{z}{2}$ (so $\gamma_{h+1} \geqslant 1-\frac{z}{2} \geqslant \frac{3}{4}$), and specifically that for every $i \leqslant h$ it holds that $\gamma_i < 1/4$. The cut that is induced by OPT on the input graph of the ith iteration has a separation weight of at least $w(OPT) - \Phi^2\cdot\sum_{j=1}^{i-1}\gamma_j$, and its cost is at most B. Hence, by the correctness of ALG,

$$\frac{c(C_i)}{\gamma_i\cdot(1-\sum_{j=1}^{i}\gamma_j)\cdot\Phi^2} \leqslant \alpha\cdot\frac{B}{\Phi^2\cdot(z-\sum_{j=1}^{i-1}\gamma_j)},$$

which means that

$$\gamma_i\cdot\left(1-\sum_{j=1}^{i}\gamma_j\right) \geqslant \frac{c(C_i)\cdot(z-\sum_{j=1}^{i-1}\gamma_j)}{\alpha\cdot B}.$$

By summing over all the iterations we have:

$$\begin{aligned} w(C) &= \sum_{i=1}^{h}\gamma_i\cdot\left(1-\sum_{j=1}^{i}\gamma_j\right)\cdot\Phi^2 \geqslant \frac{\Phi^2}{\alpha\cdot B}\cdot\sum_{i=1}^{h}c(C_i)\cdot\left(z-\sum_{j=1}^{i-1}\gamma_j\right) \\ &= \frac{\Phi^2}{\alpha\cdot B}\cdot\left(c(C)\cdot z-\sum_{i=1}^{h}c(C_i)\cdot\sum_{j=1}^{i-1}\gamma_j\right) \geqslant \frac{\Phi^2\cdot c(C)}{\alpha\cdot B}\cdot\left(z-\sum_{i=1}^{h-1}\gamma_i\right) \geqslant \frac{z\Phi^2\cdot c(C)}{2\alpha\cdot B}. \end{aligned}$$

We distinguish between the different terminations conditions. If it is the case that the solution cost exceeds the budget B, then $w(C) \geqslant \frac{w(OPT)}{2\alpha}$ and the theorem follows. Lastly, consider the case that the last connected component contains only one vertex with a positive weight. Since it follows from Inequality (9) that $z \leqslant 1-\gamma_{h+1}$, we have that

$$w(C) \geqslant \Phi^2\cdot\gamma_{h+1}\cdot(1-\gamma_{h+1}) \geqslant \Phi^2\cdot\gamma_{h+1}\cdot z \geqslant z\cdot\Phi^2\cdot\frac{3}{4} = \frac{w(OPT)}{4/3}$$

and the theorem follows. □

References

[1] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, M. Yannakakis, The complexity of multiterminal cuts, SIAM Journal on Computing 23 (4) (1994) 864–894. Preliminary version appeared in: Proc. of the 24th ACM Symposium on Theory of Computing, 1992, pp. 241–251.

[2] G. Călinescu, H.J. Karloff, Y. Rabani, An improved approximation algorithm for MULTIWAY CUT, Journal of Computer and Systems Sciences 60 (3) (2000) 564–574.

[3] D.R. Karger, P. Klein, C. Stein, M. Thorup, N.E. Young, Rounding algorithms for a geometric embedding of minimum multiway cut, in: Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC'99), ACM Press, New York, 1999, pp. 668–678.

[4] T. Aura, M. Bishop, D. Sniegowski, Analyzing single-server network inhibition, in: Proceedings of the 13th IEEE Computer Security Foundations Workshop (CSFW'00), IEEE Computer Society Press, 2000, pp. 108–117.

[5] G.N. Frederickson, R. Solis-Oba, Increasing the weight of minimum spanning trees, Journal of Algorithms 33 (2) (1999) 244–266. Preliminary version appeared in: Proc. of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms, 1996, pp. 539–546.

[6] S. Arora, S. Rao, U. Vazirani, Expander flows, geometric embeddings and graph partitioning, in: Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC'04), ACM Press, New York, 2004, pp. 222–231.

[7] Y. Aumann, Y. Rabani, An O($\log k$) approximate min-cut max-flow theorem and approximation algorithm, SIAM Journal on Computing 27 (1) (1998) 291–301.

[8] N. Linial, E. London, Y. Rabinovich, The geometry of graphs and some of its algorithmic applications, Combinatorica 15 (1995) 215–245.

[9] S. Chawla, A. Gupta, H. Räcke, Approximations for generalized sparsest cut and embeddings of L_2 into L_1, in: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'05), 2005.

[10] S. Arora, J.R. Lee, A. Naor, Euclidean distortion and the sparsest cut, in: Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC'05), ACM Press, New York, 2005, pp. 553–562.

[11] S. Khuller, A. Moss, J.S. Naor, The budgeted maximum coverage problem, Information Processing Letters 70 (1) (1999) 39–45.

[12] H. Räcke, Minimizing congestion in general networks, in: Proceedings of the 43rd Annual Symposium on Foundations of Computer Science (FOCS'02), IEEE Computer Society, 2002, pp. 43–52.

[13] C. Harrelson, K. Hildrum, S. Rao, A polynomial-time tree decomposition to minimize congestion, in: Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'03), ACM Press, New York, 2003, pp. 34–43.

[14] R. Gomory, T. Hu, Multi-terminal network flows, Journal of the SIAM 9 (1961) 551–570.

[15] H. Saran, V.V. Vazirani, Finding k-cuts within twice the optimal, SIAM Journal on Computing 24 (1) (1995) 101–108.

[16] K. Jain, V.V. Vazirani, Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation, Journal of the ACM 48 (2) (2001) 274–296.

[17] J.S. Naor, H. Shachnai, T. Tamir, Real-time scheduling with a budget, in: J.C.M. Baeten, J.K. Lenstra, J. Parrow, G.J. Woeginger (Eds.), Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP'03), in: Lecture Notes in Computer Science, vol. 2719, Springer, Berlin, 2003, pp. 1123–1137.

[18] R.V. Vohra, N.G. Hall, A probabilistic analysis of the maximal covering location problem, Discrete Applied Mathematics 43 (2) (1993) 175–183.

[19] L. Wolsey, Maximizing real-valued submodular functions: Primal and dual heuristics for location problems, Mathematics of Operations Research 7 (1982) 410–425.

[20] M. Sviridenko, A note on maximizing a submodular set function subject to a knapsack constraint, Operations Research Letters 32 (1) (2004) 41–43.

[21] A. Hayrapetyan, D. Kempe, M. Pál, Z. Svitkina, Unbalanced graph cuts, in: G.S. Brodal, S. Leonardi (Eds.), Proceedings of the 13th Annual European Symposium on Algorithms (ESA'05), in: Lecture Notes in Computer Science, vol. 3669, Springer, Berlin, 2005, pp. 191–202.

[22] Z. Svitkina, É. Tardos, Min-max multiway cut, in: K. Jansen, S. Khanna, J.D.P. Rolim, D. Ron (Eds.), Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX-RANDOM'04), in: Lecture Notes in Computer Science, vol. 3122, Springer, Berlin, 2004, pp. 207–218.

[23] O.H. Ibarra, C.E. Kim, Fast approximation algorithms for the knapsack and sum of subset problems, Journal of the ACM 22 (4) (1975) 463–468.

[24] J.-H. Lin, J.S. Vitter, ε-approximations with minimum packing constraint violation (extended abstract), in: Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC'92), ACM Press, New York, 1992, pp. 771–782.

[25] N. Garg, V.V. Vazirani, M. Yannakakis, Primal-dual approximation algorithms for integral flow and multicut in trees, Algorithmica 18 (1) (1997) 3–20.

[26] V.V. Vazirani, Approximation Algorithms, first ed., Springer, Berlin, 2001.